```
(FILE 'HOME' ENTERED AT 12:14:17 ON 17 JUL 2001)

     FILE 'USPATFULL' ENTERED AT 12:16:36 ON 17 JUL 2001
L1          4303 S ENCRYPT?(5A)KEY#
L2          2716 S DECRYPT?(5A)KEY#
L3          2517 S L1 AND L2
L4          2185 S TAMPER(3W)RESISTANT
L5           153 S L3 AND L4
L6           353 S RE(3W)ENCRYPT? OR REENCRYPT?
L7            43 S L6 AND L5
L8          1934 S ENCRYPT?(10A)(DIGITAL OR ELECTRONIC(3A)CONTENT)
L9           860 S DECRYPT?(10A)(DIGITAL OR ELECTRONIC(3A)CONTENT)
L10          723 S L8 AND L9
L11           14 S (RE(3W)ENCRYPT? OR REENCRYPT?)(10A)(DIGITAL OR
ELECTRONIC(3A)
L12           13 S L10 AND L11
L13            3 S L12 AND L4
L14           70 S L8 AND L9 AND L4
L15         1421 S ENCRYP?(5A)(CONTENT OR APPLICATION OR SOFTWARE#)
L16          637 S DECRYP?(5A)(CONTENT OR APPLICATION OR SOFTWARE#)
L17          466 S L15 AND L16
L18           56 S L17 AND L4
```

L19 ANSWER 18 OF 22 USPATFULL

SUMM . . . relates to distributed and other operating systems, environments and architectures. It also generally relates to secure architectures, including, for example, **tamper-resistant** hardware-based processors, that can be used to establish security at each node of a distributed system.

SUMM . . . impede, interference with and/or observation of, important rights related transactions and processes. VDE, in its preferred embodiment, uses special purpose **tamper resistant** Secure Processing Units (SPUs) to help provide a high level of security for VDE processes and information storage and communication.

SUMM . . . into, many separate computers and/or other electronic appliances. These appliances typically include a secure subsystem that can enable control of **content** use such as displaying, **encrypting**, **decrypting**, printing, copying, saving, extracting, embedding, distributing, auditing usage, etc. The secure subsystem in the preferred embodiment comprises one or more. . .

SUMM . . . processes securely controlled in VDE participant user installations (nodes). VDE installations, in the preferred embodiment, may include both software and **tamper resistant** hardware semiconductor elements. Such a semiconductor arrangement comprises, at least in part, special purpose circuitry that has been designed to. . .

SUMM . . . stored information to ensure against substitution and tampering) and distributed content (to, for many content applications, employ one or more **content encryption** keys that are unique to the specific VDE installation and/or user), private key techniques such as triple DES to **encrypt content**, public key techniques such as RSA to protect communications and to provide the benefits of digital signature and authentication to. . .

SUMM . . . extraction or copying. Fingerprinting normally is embedded into unencrypted electronic content or control information, though it can be embedded into **encrypted content** and later placed in unencrypted content in a secure VDE installation sub-system as the **encrypted content** carrying the fingerprinting information is **decrypted**. Electronic information, such as the **content** of a VDE container, may be fingerprinted as it leaves a network (such as Internet) location bound for a receiving. . . encrypted as it leaves the repository. Fingerprinting would preferably take place as the content leaves the repository, but before the **encryption** step. **Encrypted** repository **content** can be **decrypted**, for example in a secure VDE sub-system, fingerprint information can be inserted, and then the **content** can be **re-encrypted** for transmission. Embedding identification information of the intended recipient user and/or VDE installation into content as it leaves, for example,. . . information. Fingerprint information identifying a receiving party and/or VDE installation can be embedded into a VDE object before, or during, **decryption**, replication, or communication of VDE **content** objects to receivers. Fingerprinting electronic **content** before it is **encrypted** for transfer to a customer or other user provides information that can be very useful for identifying who received certain. . .

SUMM . . . video monitor or other display device, and such device would be designed, to the extent commercially practical, to be as **tamper resistant** as reasonable. As another example, embedding a VDE

hardware module into an I/O peripheral may have certain advantages from the. . .

SUMM . . . be allowed by "senior" participants (by content creators, for example) to require a method which prohibits end-users from electronically saving **decrypted content**, a provider of credit for VDE transactions might require an audit method that records the time of an electronic purchase,. . .

SUMM . . . secure subsystems and the effectiveness of the SPU hardware security architecture, software security techniques when an SPU is emulated in **software**, and the **encryption** algorithm(s) and keys that are employed for securing content, control information, communications, and access to VDE node (VDE installation) secure. . .

SUMM . . . activities is required. Such a trusted environment may be created through the use of certain control software, one or more **tamper resistant** hardware modules such as a semiconductor or semiconductor chipset (including, for example, a **tamper resistant** hardware electronic appliance peripheral device), for use within, and/or operatively connected to, an electronic appliance. With the present invention, the trustedness of a hardware SPU can be enhanced by enclosing some or all of its hardware elements within **tamper resistant** packaging and/or by employing other tamper resisting techniques (e.g. microfusing and/or thin wire detection techniques). A trusted environment of the present invention implemented, in part, through the use of **tamper resistant** semiconductor design, contains control logic, such as a microprocessor, that securely executes VDE processes.

SUMM . . . is in contrast to the preferred embodiment wherein a trusted environment is created using a combination of one or more **tamper resistant** semiconductors that are not part of said primary control logic. In either embodiment, certain control information (software and parameter data). . .

DETD SPU 500 is enclosed within and protected by a "**tamper resistant** security barrier" 502. Security barrier 502 separates the secure environment 503 from the rest of the world. It prevents information. . . secure conditions. Barrier 502 also controls external access to secure resources, processes and information within SPU 500. In one example, **tamper resistant** security barrier 502 is formed by security features such as "encryption," and hardware that detects tampering and/or destroys sensitive information. . .

DETD As shown in FIGS. 6 and 9, SPU 500 may be surrounded by a **tamper -resistant** hardware security barrier 502. Part of this security barrier 502 is formed by a plastic or other package in which. . .

DETD . . . (or destroys other parts of the SPU) when tampering is detected. These and other hardware-based physical security techniques contribute to **tamper-resistant** hardware security barrier 502.

DETD . . . handles the most security sensitive aspects of the operation of electronic appliance 600. For example, microprocessor 520 may manage VDE **decrypting**, **encrypting**, certain **content** and/or appliance usage control information, keeping track of usage of VDE secured content, and other VDE usage control related functions.

DETD In the preferred embodiment, SPU 500 includes a real time clock circuit ("RTC") 528 that serves as a reliable, **tamper resistant** time base for the SPU. RTC 528 keeps track of time of day and date (e.g., month, day and year). . .

DETD . . . and efficiently encrypting and/or decrypting data. In some implementations, the encrypt/decrypt functions may be performed instead by microprocessor 520 under **software** control, but providing special purpose **encrypt**/decrypt hardware engine 522 will, in general, provide increased performance. Microprocessor 520 may, if

desired, comprise a combination of processor circuitry. . .

DETD . . . 662 of electronic appliance 600 in which SPU 500 resides. The symmetric key encryption/decryption circuit may also be used for **encrypting** and **decrypting content** stored within VDE objects 300.

DETD . . . loaded into the memory and communicated to and decrypted within SPU 500 prior to execution. Such control programs may be **re-encrypted** and communicated back to external memory where they may be stored for later execution by SPU 500. "Kernel" programs and/or. . .

DETD . . . in such multi-package versions, it may be necessary to enclose all the packages and their interconnections in an external physical **tamper-resistant** barrier.

DETD . . . the SPU chip 2660 by a private memory bus 2661, and all three such components would be contained within hardware **tamper-resistant** barrier 502.

DETD . . . integrated circuit package, and/or on a single silicon die. This could reduce packaging complexity and/or simplify establishment of the hardware **tamper-resistant** barrier 502.

DETD . . . is a secure processing environment provided at least in part by an SPU 500. Thus, SPU 500 provides the hardware **tamper-resistant** barrier 503 surrounding SPE 503. SPE 503 provided by the preferred embodiment is preferably:

DETD HPEs 655 may (as shown in FIG. 10) be provided with a software-based **tamper resistant** barrier 674 that makes them more secure. Such a software-based **tamper resistant** barrier 674 may be created by software executing on general-purpose CPU 654. Such a "secure" HPE 655 can be used. . .

DETD The software-based **tamper resistant** barrier 674 provided by HPE 655 may be provided, for example, by: introducing time checks and/or code modifications to complicate. . . of electronic appliance 600 to "protect" the operation of HPE 655 from other processes, functions, etc. Although such a software-based **tamper resistant** barrier 674 may provide a fair degree of security, it typically will not be as secure as the hardware-based **tamper resistant** barrier 502 provided (at least in part) by SPU 500. Because security may be better/more effectively enforced with the assistance. . .

DETD . . . a series of events to SPE 503 to create one or more PERCs 808, public headers, private headers, and to **encrypt content**, all for storage in the new object 300 (or within secure database 610 within records associated with the new object).

DETD . . . to the object switch for incorporation into the object. Such information provided by SPE 503 may include, in addition to **encrypted content** or other information, one or more PERCs 808, one or more method cores 1000', one or more load modules 1100,. . .

DETD . . . other information as specified by object configuration file 1240. Container manager 764 may then insert into the container 302 the **content** or other information (as **encrypted** by SPE 503) to be included in the new object. Container manager 764 may also insert appropriate permissions, rules and/or. . .

DETD . . . encryption/decryption engine 522 within SPU 500. Those encryption/decryption technologies not supported by SPU encrypt/decrypt engine 522 may be provided by **encrypt/decrypt** engine 556 in **software**. The primary bulk **encryption /decryption** load modules preferably are loaded at all times, and the load modules necessary for other algorithms are preferably paged in. . .

DETD . . . (user-specified) processes and data.

Process Method Cores Provide a mechanism to

Elements relate events to control mechanisms and permissions.

Load Modules ("LMs") Secure (**tamper-resistant**) executable code.

Method Data Elements ("MDEs") Independently deliverable data structures used to control/customize methods.

Data Structures Permissions Records Permissions to use

. . .

DETD . . . the object or its contents. The permission records 808 for an object may include key block(s) 810, which may store **decryption** keys for accessing the **content** of the **encrypted content** stored within the object 300.

DETD . . . because VDE objects may contain data that can be electronically copied outside the confines of a VDE node. If the **content** is **encrypted**, the copies will also be encrypted and the copier cannot gain access to the **content** unless she has the appropriate **decryption** key(s). For objects in which maintaining security is particularly important, the permission records 808 and key blocks 810 will frequently. . .

DETD . . . time the record is updated (or after a certain one or more events). In this event, the updated record is **re-encrypted** with new one or more keys. Alternately, one or more of the keys used to encrypt and decrypt management information. . . site information, absolute points in time, and/or duration of time related to a subset of activities related to using or **decrypting** VDE secured **content** or the use of the VDE system.

DETD . . . keys contained in the separate permissions record 808. The data blocks 812 contain content (information or administrative) that may be **encrypted** using one or more **content** keys also provided in permissions record 808.

DETD Since the **content** of the traveling object is **encrypted**, it can be used only under authorized circumstances unless the traveling object private header key used with the object is. . .

DETD . . . keys used with "rights" (for encoding and/or decoding audit trails, for example). It may contain the keys for the object **content** or keys to **decrypt** portions of the object that contain other keys that then can be used to **decrypt** the **content** of the object. Usage of the keys is controlled by the control sets 914 in the same "right" 906 within. . .

DETD . . . embodiment. Such right keys 912 may include, for example, decryption keys that enable a method specified by PERC 808 to **decrypt content** for release by a VDE node to an end user. These right keys 912 are, in the preferred embodiment, unique. . .

DETD . . . the element may be automatically rejected, error corrected, etc. Assuming the element is found to have integrity, SPE 503 may **re-encrypt** the information (block 1080) using a new key for example (see FIG. 37 discussion below). In the same process step. . .

DETD . . . At some point, this internal list may fill up. At this point, SPE 503 may call a maintenance routine that **re-encrypts** items within secure database 610 containing changed information. Some or all of the items within the data structure containing changed information may be read in, decrypted, and then **re-encrypted** with the same key. These items may then

be issued the same key identification number. The items may then be. . .

DETD . . . in the memory exceeding a predetermined number, a timer has expired, etc.), then the preferred embodiment handles the situation by **re-encrypting** other records with secure database 610 with the same new key in order to reduce the number of (or change). . . selected, and all secure database items encrypted using the old key(s) are read and decrypted. These records may now be **re-encrypted** using the new key that was generated at block 1086 for the new record (block 1094). The old key(s) used. . . records within the secure database 610 that were encrypted using the old key(s) have been read by block 1092 and **re-encrypted** by block 1904 using the new key. All records encrypted (or **re-encrypted**) using the new key may now be stored in secure database 610 (block 1098). If decision block 1090 determines there. . .

DETD . . . be provided by additional one or more integrated circuits that can be contained within a secure enclosure, such as a **tamper resistant** metal container or some form of a chip pack containing multiple integrated circuit components, and which impedes and/or evidences tampering. . .

DETD . . . used to handle error conditions. A DECRYPT method is used to decrypt encrypted information. An ENCRYPT method is used to **encrypt** information. A DESTROY **content** method is used to destroy the ability to access specific content within a container. An INFORMATION method is used to. . .

DETD . . . and a BUDGET method 1660. In the preferred embodiment, READ control method 1652 may request methods to fingerprint and/or obscure **content** before releasing the **decrypted content** .

DETD Referring to FIG. 50f, the READ control method 1652 must determine which key to use to **decrypt content** if it is going to release **decrypted content** to the user (block 1758). READ control method 1652 may make this key determination based, in part, upon the PERC. . . 808 for the object (block 1760). READ control method 1652 may then call an ACCESS method to actually obtain the **encrypted content** to be **decrypted** (block 1762). The **content** is then **decrypted** using the key determined by block 1758 (block 1764). READ control method 1652 may then determine whether a "fingerprint" is. . . call the FINGERPRINT method (block 1768). Otherwise, READ control method 1652 may determine whether it is desired to obscure the **decrypted content** (decision block 1770). If so, READ control method 1652 may call an OBSCURE method to perform this function (block 1772).. . .

DETD . . . be metered, billed and budgeted. As shown in FIG. 51, the end result of WRITE method 1780 is typically to **encrypt content**, update the container table of contents and related information to reflect the new content, and write the content to the. . .

DETD . . . determines (based on the PERC for the object and user and an optional algorithm) which key should be used to **encrypt** the **content** before it is written to the container (blocks 1894, 1896). CONTROL method 1782 then **encrypts** the **content** (block 1898) possibly by calling an **ENCRYPT** method, and writes the **encrypted content** to the object (block 1900). CONTROL method 1782 may then update the table of contents (and related information) for the. . .

DETD . . . reads the ACCESS method MDE from the secure database, reads it in accordance with the ACCESS method DTD, and loads **encrypted**

content source and routing information based on the MDE (blocks 2010, 2012). This source and routing information specifies the location of the **encrypted content**. ACCESS method 2000 then determines whether a connection to the content is available (decision block 2014). This "connection" could be,. . . failure indication (termination point 2018). If the open connection succeeds, on the other hand, then ACCESS method 2000 obtains the **encrypted content** (block 2020). ACCESS method 2000 then writes an ACCESS audit trail if required to the secure database ACCESS method Audit. . .

DETD . . . 2030 in the preferred embodiment obtains or derives a decryption key from an appropriate PERC 808, and uses it to **decrypt** a block of **encrypted content**. **DECRYPT** method 2030 is passed a block of **encrypted content** or a pointer to where the encrypted block is stored. DECRYPT 2030 selects a key number from a key block (block 2032). For security purposes, a **content** object may be **encrypted** with more than one key. For example, a movie may have the first 10 minutes encrypted using a first key,. . . a "key block." The selection process involves determining the correct key to use from the key block in order to **decrypt** the **content**. The process for this selection is similar to the process used by EVENT methods to map events into atomic element. . . "seed") from a PERC (blocks 2034, 2036). This key information may be the actual decryption key to be used to **decrypt** the **content**, or it may be information from which the decryption key may be at least in part derived or calculated. If. . .

DETD . . . stored in the key block of PERC 808 so that DECRYPT method 2030 may access the key in order to **decrypt** the **content** stored in the **content** object. **ENCRYPT** method 2050 then accesses the appropriate PERC to obtain, derive and/or compute an encryption key to be used to encrypt. . .

DETD . . . block 2072), then CONTENT method 2070 may simply read this static value content information from the object (block 2074), optionally **decrypt**, and release this **content** description (block 2076). If, on the other hand, CONTENT method 2070 must derive the synopsis/content description from the secure object. . .

DETD . . . released from a content object 300. However, it could also be performed upon distribution of an object to "mark" the **content** while still in **encrypted** form. For example, a network-based object repository could embed fingerprints 2161 into the content of an object before transmitting the. . .

DETD . . . (a) security, and (b) processing time and/or resources. Since a hardware-based PPE encrypt/decrypt engine 522 may provide faster processing than **software**-based **encryption**/ **decryption**, the hardware-based approach may, in general, allow use of longer keys.

DETD . . . older PPEs to maintain compatibility with newer PPEs and/or newly released VDE objects 300 and other VDE-protected information. For example, **software encryption/decryption** algorithms may be downloaded into PPE 650 at any time to supplement the hardware-based functionality of encrypt/decrypt engine 522 by. . .

DETD . . . content user will be different from the content creator's result. If the result is used as a symmetrical key for **encryption** by the **content** creator, the **content** user will not be able to **decrypt** unless the **content** user's result is the same as the result of the content creator.

DETD . . . convoluted key 2862(z), which may then be used to generate the content key 2863 in the object's PERC 808. To **decrypt** the object's **content**, the user site may use each of its sequence of convolution keys 2862(a-e) to attempt to generate the master content.

DETD 1) A creator makes a "true" key, and **encrypts content** with it.

DETD . . . If the supplied time and/or other information is "wrong," the convolution function will not yield the "true" key, and therefore **content** cannot be **decrypted**.

DETD . . . content object 850. The private body key(s) 810 may then be extracted from the PERC 808 and used by the **content decryption** process 2845 to make the **content** available outside the PPE 650. In addition, the database key(s) 2817 may be used by the encryption process 2844 to. . .

DETD . . . dependent on key information shared between PPEs 650. They are preferably generated by the PPE 650 at the time the **content** is **encrypted**. They may incorporate time as a component to "age" them. They are received in permissions records 808, and their usage. . .

DETD . . . database records and current audit "roll ups" stored in both PPE 650 and externally. Then, the backup process decrypts and **re-encrypts** this information using a new set of generated keys. These keys, the time of the backup, and other appropriate information. . .

DETD . . . record without knowing the encryption key (either in the part representing the data or the part representing the seal), the **decrypted content** will be different, and the **decrypted** check value will not match the digest computed from the record's data. Even though the hash algorithm is known, it. . .

DETD . . . cryptographic algorithm that are used to implement VDE 100 are assumed to be safe (cryptographically strong). These include the secret-key **encryption** of **content**, public-key signatures for integrity verification, public-key encryption for privacy between PPEs 650 or between a PPE and a VDE administrator,. . .

DETD . . . "aged" by including a time component. Time is convoluted with the stored values to derive the "true keys" needed to **decrypt content**. If this process is also compromised, then object content or methods would be revealed. Since a backup of secured database. . .

DETD . . . the shared external communications keys, site ID and time. Knowledge of PPE 650 internal details would be necessary to further **decrypt** the **content** of administrative objects.

DETD If a content key becomes compromised, the portion of the **content encrypted** with the key is also compromised until the key "ages" and expires. If the "aging" process for that key also becomes compromised, then the breach becomes permanent. If multiple levels of encryption are used, or portions of the **content** are **encrypted** with different keys, learning a single key would be insufficient to release some or all of the content.

DETD . . . may use a software-based protected processing environment 650 (such as a "host event processing environment" (HPE) 655) providing a software-based **tamper resistant** barrier 674. Software-based **tamper resistant** barrier 674 may be created by software executing on a general-purpose CPU. Various software protection techniques may be used to construct and/or provide software-based **tamper resistant** barrier 674.

DETD . . . connection with PPE 650 apply to a software-based PPE. An important threat to be countered with respect to a software-based **tamper resistant** barrier 674 is an attack based on a distributable computer program that can defeat the **tamper resistant** barrier wherever the program is run. Since a software-based **tamper resistant** barrier 674 typically will not be as secure as a hardware-based **tamper resistant** barrier 502, it is useful to explore example steps and procedures a "cracker" might use to "`crack` a software"-based **tamper resistant** barrier.

DETD FIGS. 67A and 67B show example "cracking" techniques a "cracker" might use to attack software-based **tamper resistant** barrier 674.

DETD Referring to FIG. 67A, the software used to create **tamper resistant** barrier 674 may be distributed, for example, on a storage medium 3370 such as a floppy diskette or optical disk. . . .

DETD . . . execute the operational materials 3472 from its hard disk 3376 to provide software-based protected processing environment 650 and associated software-based **tamper resistant** barrier 672.

DETD No software-only **tamper resistant** barrier 674 can be wholly effective against all of these threats. A sufficiently powerful dynamic analysis (such as one employing. . . . time consuming--increasing the "work factor" to a point where it may become commercially unfeasible to attempt to "crack" a software-based **tamper resistant** barrier 674.

DETD Example Techniques for Forming Software-Based **Tamper Resistant** Barrier

DETD Various software protection techniques detailed above in connection with FIG. 10 may provide software-based **tamper resistant** barrier 674 within a software-only and/or hybrid software/hardware protected processing environment 650. The following is an elaboration on those above-described. . . .

DETD In general, the software-based **tamper resistant** barrier 674 may establish "trust" primarily through uniqueness and complexity. In particular, uniqueness and customization complicate the ability of an. . . .

DETD make it harder for an attacker to create a software program(s) that will defeat the **tamper-resistant** barrier 674 of multiple PPE instances;

DETD In addition, the overall software-based **tamper resistant** barrier 674 and associated PPE system is sufficiently complex so that it is difficult to tamper with a part of. . . .

DETD **encrypted software** distribution,

DETD . . . operational materials 3472 may provide executable code and associated data structures for providing protected processing environment 650 and associated software-based **tamper resistant** barrier 674.

DETD . . . keys so that compromise of one key exposes only a subset of the software distribution to unwanted disclosure. The only non-**encrypted** part of the **software** distribution in plaintext is that portion 3470C of installation materials 3470 used to establish initial contact with the registry 3476.

DETD uniquely **encrypting** the installed **software**,

DETD The installation process for the operational **software** may involve **decrypting** its distribution (which may be the same for all end users) and modifying it to encode the specific locations where. . . .

DETD . . . the next program segment dynamically (FIG. 69K, block 3460. The code may be decrypted dynamically when it is needed, then **re-encrypted** or overwritten and discarded when not in use. This mechanism increases the tamper-resistance of the executable code--thus providing additional tamper. . . .

DETD Load Module Dynamic Decryption & **Re-Encryption**

DETD The executing operational materials 3472 may decrypt load module 1100 code dynamically as needed, and **re-encrypt** it or otherwise render it inscrutable when not in use (FIG. 69L, block 3580). In accordance with this technique, load module executable code and/or data is decrypted dynamically when it is needed, then **re-encrypted** or destroyed when not in use. In addition, the

location of executing load modules 1100 may be varied randomly to. . .

DETD . . . restoration processes from being misused to allow such replay attacks. Such checks may identify incomplete or erroneous attempts to subvert **tamper resistant** barrier 672. Great care must be taken to ensure that these checks do not trigger as a result of execution. . .

DETD . . . FIG. 69N). This verification may counter attacks that might, for example, attempt to trick PPE 650 access methods into releasing **content** that has been **decrypted** but not electronically fingerprinted. Executing operational materials 3472 may also include numerous internal consistency checks to prevent substitution (replay) of. . .

DETD . . . 2631 connecting the electronic appliances can be VDE-protected (e.g., it may be packaged in the form of VDE administrative and/or **content** objects and **encrypted** as discussed above). One of the consequences of this arrangement is that an eavesdropper who taps into communications path 2631. . .

DETD tampered with, as a precondition to creating and delivering the **encrypted** printable **content** 3902. The **decryption** program 3900 could also

DETD **content** streams 3902. The **decryption** program 3904 could destroy itself

DETD after printing one or more **encrypted content** streams 3902 to protect

DETD . . . There are similarly different patterns for other characters in fonts 3910a-3910z, shown as 3913ab-3913zb, etc., permitting a more efficient and/or **tamper-resistant** encoding of fingerprint information.

DETD Housing 2602 may be **tamper resistant**. (See discussion above relating to tamper resistance of SPU barrier 502.)

DETD One possible "stand alone" configuration for **tamper-resistant**, portable appliance 2600 arrangements includes a **tamper-resistant** package (housing 2602) containing one or more processors (500, 2616) and/or other computing devices and/or other control logic, along with. . . would also possess the ability to store at least a portion of permission, method, and/or key information securely within said **tamper resistant** portable housing 2602 on non-volatile memory.

DETD . . . of files, and so on, without limitation. The authoring process may encapsulate content generated by the author in an object, **encrypt** the **content** with one or more keys, and append one or more methods to define parameters of allowed use and/or required auditing. . .

DETD An important aspect of adding or modifying **content** is the choice of **encryption/decryption** keys and/or other relevant aspects of securing new or altered content. The provider may specify in their method(s) associated with. . .

DETD . . . but not necessarily identical. For example, key updates for a budget may control encryption of a billing trail, rather than **decryption** of object **content**. The billing trail for a budget is in all respects a method event trail. In one embodiment, this trail must. . .

DETD . . . information exchanged under privilege, properly controlled, and not inappropriately released and/or otherwise used. The materials (content) stored in a VDE **content** container object will normally be **encrypted**. Thus wrapped, a VDE object may be distributed to the recipient without fear of unauthorized access and/or other use. The. . .

DETD . . . a desirable location(s) to make their content available for easy access by users. If a repository, such as CompuServe, stores **content** in non-**encrypted** (plaintext) form, it may **encrypt** "outgoing" **content** on an "as needed" basis

through placing such content in VDE content containers with desired control information, and may employ. . .

DETD how content that is stored at and/or passed through the repository should be shipped (including any container criteria, **encryption** requirements, transaction requirements related to **content** transmissions, other control criteria, etc.)

DETD . . . performed by the repository as an aspect of delivering content.

For example, author 3306A may have required that the repository **encrypt** each copy of shipped **content** using a different **encryption** key or keys in order to help maintain greater protection for **content** (e.g. in case an **encryption** key was "cracked" or inadvertently disclosed, the "damage" could be limited to the portion(s) of that specific copy of a certain **content** deliverable). In another example, **encryption** functions may include the need to use entirely different encryption algorithms and/or techniques in order to fulfill circumstantial requirements (e.g.. . .

DETD . . . container may be relocated through the use of VDE sub-system secure processes which may, for example, continue to maintain relocated **content** as **encrypted** or otherwise protected (e.g. by secure **tamper resistant** barrier 502) during a relocation/embedding process.

DETD . . . to embed their content container in another container while maintaining a requirement that creator B receive $0.50 per kilobyte of **content decrypted**, (c) have no restrictions on the number of enabling control information sets that may be generated for users and/or user/distributors,. . .

DETD . . . this example does not force a model including "rental" of rights, but rather bases payment amounts on the quantity of **content decrypted** by a user or user/distributor. In this example, distributor A may use VDE to negotiate with creator B to include. . .

DETD . . . set of control information D.sub.A (C.sub.B) may include one or

more meter methods that record the number of bytes of **content** from creator B's container **decrypted** by user A (in order to allow correct calculation of amounts owed by distributor A to creator B for user. . . For example, user/distributor A may receive control information C.sub.B that includes a requirement that user/distributor A pay creator B for **content decrypted** by user/distributor A (and any participant receiving distributed and/or redistributed control information from user/distributor A) at the rate of $0.50. . .

DETD . . . The Audio Library 3404 has established similar controls that match its business model. The Internet Repository 3406 VDE containerizes, including **encrypts**, selected object **content** as it streams out of the Repository in response to an online, user request to download an object. The Repository 3406 may fingerprint the identification of the receiving VDE installation into its **content** prior to **encryption** and communication to a publisher, and may further require user identification fingerprinting of their **content** when **decrypted** by said Publisher or other **content** user.

CLM What is claimed is:

. . . processing unit comprising a CPU, microprocessor or microcontroller and components designed to perform security-related functions, said components including: a secure, **tamper-resistant** barrier operating to render unauthorized interference with or access to the contents or operations of the secure processing unit more. . .

. . . processing unit comprising a CPU microprocessor or microcontroller and components designed to perform security-related functions, said components including: a secure, **tamper-resistant** barrier operating to render unauthorized interference with or access to the contents or operations of the secure processing unit more. . .

. . . processing unit comprising a CPU, microprocessor or microcontroller and components designed to perform security-related functions, said components including: a secure, **tamper-resistant** barrier operating to render unauthorized interference with or access to the contents or operations of the secure processing unit more. . .

. . . processing unit comprising a CPU, microprocessor or microcontroller and components designed to perform security-related functions, said components including: a secure, **tamper-resistant** barrier operating to render unauthorized interference with or access to the contents or operations of the secure processing unit more. . .

. . . processing unit comprising a CPU, microprocessor or microcontroller and components designed to perform security-related functions, said components including: a secure, **tamper-resistant** barrier operating to render unauthorized interference with or access to the contents or operations of the secure processing unit more. . .

. . . central processing unit mass storage operatively connected to said central processing unit and said main memory; said main memory storing **tamper resistant** software designed to be loaded into said main memory and executed by said central processing unit said **tamper resistant** software comprising: programming which uses at least one confounding algorithm to create critical values required for correct operation of at. . .

. . . central processing unit mass storage operatively connected to said central processing unit and said main memory; said main memory storing **tamper resistant** software designed to be loaded into said main memory and executed by said central processing unit, said **tamper resistant** software comprising: programming which uses a multiplicity of confounding algorithms to create critical values required for correct operation of at. . .

. . . central processing unit mass storage operatively connected to said central processing unit and said main memory; said main memory storing **tamper resistant** software designed to be loaded into said main memory and executed by said central processing unit, said **tamper resistant** software comprising: programming which uses at least one confounding algorithm to create critical values required for correct operation of at. . .

. . . central processing unit; mass storage operatively connected to said central processing unit and said main memory; said mass storage storing **tamper resistant** software designed to be loaded into said main memory and executed by said central processing unit, said **tamper resistant** software comprising: a multiplicity of software identifier storage locations, a first of said software identifier storage locations containing an integrity value embedded in said **tamper resistant** software at least in part for the purpose of identifying said **tamper resistant** software; and a second of said software identifier storage locations containing a string of bits of the same length as said integrity value, said string of bits embedded in said **tamper resistant** software at least in part for the purpose of obscuring said integrity value.

. . . environment as in claim 134, said second host processing environment further comprising, location encryption programming containing programming which records and **encrypts** the location of said first **software** identifier storage location, said encryption taking place using a location cryptographic key.

. . . central processing unit; mass storage operatively connected to said central processing unit and said main memory; said mass storage storing **tamper resistant** software designed to be loaded into said main memory and executed by said central processing unit, said **tamper resistant** software comprising: machine check programming which derives information from one or more aspects of said host processing environment, one or. . .

. . . central processing unit; mass storage operatively connected to said

central processing unit and said main memory; said mass storage storing
tamper resistant software designed to be loaded into
said main memory and executed by said central processing unit, said
tamper resistant software comprising: machine check
programming which derives information from one or more aspects of said
host processing environment, one or. . .

. . . central processing unit; mass storage operatively connected to said
central processing unit and said main memory; said mass storage storing
tamper resistant software designed to be loaded into
said main memory and executed by said central processing unit, said
tamper resistant software comprising: machine check
programming which derives information from one or more aspects of said
host processing environment, one or. . .

. . . central processing unit; mass storage operatively connected to said
central processing unit and said main memory; said mass storage storing
tamper resistant software designed to be loaded into
said main memory and executed by said central processing unit, said
tamper resistant software comprising: machine check
programming which derives information from one or more aspects of said
host processing environment, one or. . .

. . . in part secure, mass storage operatively connected to said central
processing unit and said main memory; said mass storage storing
tamper resistant software designed to be loaded into
said main memory and executed by said central processing unit, said
tamper resistant software comprising: database check
programming which derives information from one or more aspects of the
state of said database, one. . .

=> d 18 ibib

L19 ANSWER 18 OF 22 USPATFULL

ACCESSION NUMBER:	1999:44617 USPATFULL
TITLE:	Systems and methods for secure transaction management and electronic rights protection
INVENTOR(S):	Ginter, Karl L., Beltsville, MD, United States Shear, Victor H., Bethesda, MD, United States Sibert, W. Olin, Lexington, MA, United States Spahn, Francis J., El Cerrito, CA, United States Van Wie, David M., Sunnyvale, CA, United States
PATENT ASSIGNEE(S):	InterTrust Technologies Corp., Sunnyvale, CA, United States (U.S. corporation)

	NUMBER	KIND	DATE	
PATENT INFORMATION:	US 5892900		19990406	
APPLICATION INFO.:	US 1996-706206		19960830	(8)

DOCUMENT TYPE:	Utility
FILE SEGMENT:	Granted
PRIMARY EXAMINER:	Beausoliel, Jr., Robert W.
ASSISTANT EXAMINER:	Elisca, Pierre F.
LEGAL REPRESENTATIVE:	Nixon & Vanderhye P.C.
NUMBER OF CLAIMS:	220
EXEMPLARY CLAIM:	1
NUMBER OF DRAWINGS:	177 Drawing Figure(s); 163 Drawing Page(s)
LINE COUNT:	22540